Using SourceServer 24

SourceServer is an application that provides capabilities for coordinating source code, documentation, and resources that are used by the THINK Project Manager to build applications. SourceServer uses a database server named “Projector” to assist in managing large programming efforts of individual and group programmers. The server provides services that help track changes in source files. It allows you to safely experiment with your source code because you can still revert to any previous state or condition of your code.

Most importantly, for large projects, it provides protection for source files. Even when many people share access to a single file, all programmers involved with the project can easily coordinate their work. Changes that they make to that file are protected, automatically tracked, reported to other programmers interested in that file, and, most importantly, automatically saved by the system. Using SourceServer provides the optimum balance between ease of use, flexibility, and security.

The THINK Project Manager integrates SourceServer by using a hierarchical menu and a set of dialogs that let you perform SourceServer tasks directly from the THINK Project Manager. You can also enter SourceServer commands by typing the command line into a THINK Editor window, selecting the command, and pressing Control-Enter.

Contents

About SourceServer 469
    Overview 469
    Features 471
    Limitations 472

Using SourceServer: A Tutorial 473
    Creating and mounting a new project 473

Checking in a revision 474
Opening files in a Projector database 475
Checking out files 475
Modifying read-only files 476
Checking in your revisions 477

The Check In Dialog 478
Project pop-up menu 478
User field 478
Select files area 479
Task and comment fields 479
Keep read-only, Keep modifiable, and Delete my copy buttons 479
Touch mod date check box 479
Verify check box 480
Check In button 480
Done button 480

The Check Out Dialog 480
Select files area 481
By Name field 483
User field 483
Done button 483
Check Out button 483
Undo Checkout button 483
Info button 484
Checkout To directory field 484
Task and Comment fields 484
Read-only and Modifiable buttons 485
Branch check box 485
Touch mod date 485
Verify check box 485

The Name Revisions Dialog 486

Creating Branches 486

Components of a Projector Database 487
Projector database 488
Revision trees 489
Branches 491
Symbolic names 491

SourceServer Database Administration 491
Creating nested databases 492
Moving, renaming, and deleting projects 493
Deleting revisions 493
Removing a file from a project 494

Using the Command-Line Interface 494
SourceServer Command Descriptions 495

About SourceServer

SourceServer helps programmers (both individuals and teams) control and account for changes to all the files (documentation, source, applications, and so on) associated with a software development effort. Use SourceServer to coordinate changes among a team of programmers and to maintain a history of your revisions. When you begin work on a project, you select the Projector database you wish to work with and check out the files you need, just as you would check books out from the public library. In this case, however, SourceServer distributes both read-only and modifiable copies of its "books."

Many of the terms and concepts introduced in this section are discussed in greater detail in the section "Components of a Projector Database" later in this chapter.

Overview

During a software development effort, each team member makes numerous changes to the source and documentation files. Sometimes the changed source files are alternative versions or experimental efforts; later the team may want to discard the failed efforts and merge the best versions together. SourceServer is designed to support this process by providing an easy-to-use, powerful facility for file management that is valuable to both the individual programmer working on a small project and to teams of programmers working on complex sets of programming projects. You can use SourceServer to organize your files into SourceServer databases that can be stored locally on a hard disk, a 3.5-inch diskette, or remotely on an AppleTalk network.

At the highest level, a Projector database presents itself to the user as a flat set of files. Inside the database, each of these files becomes the root of a tree of alterations to the original file. This is called a revision tree.

When alterations are made to a file, and the file is checked back into the Projector database, a new revision of that file is added to the revision tree. A new revision becomes either the direct descendant or a branch of the original file that was checked out. Each revision tree comprises the entire historical sequence of revisions and branches of a particular file. Any of these revisions may be opened for reading only or checked out exclusively by one user for

modification. A Projector database is sometimes referred to as a project.

The figure below shows how three files might appear as three revision trees in a database. The sequentially numbered circles represent revisions. Circles with letter suffixes are branches, which may in turn sprout their own branches and subsequent revisions.

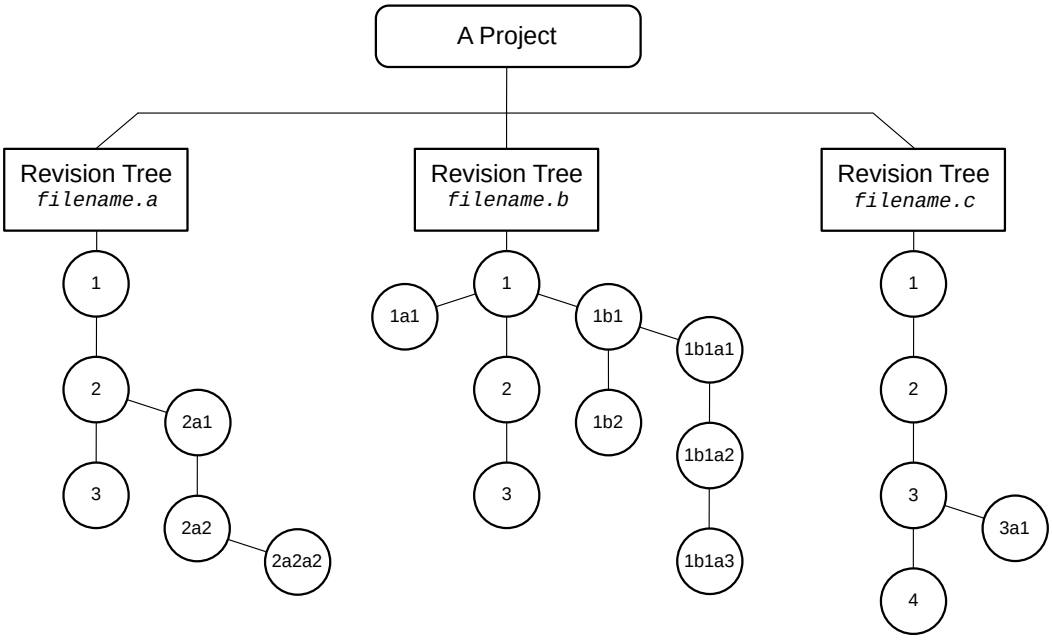

Figure 24-1 Project structure

When you check out a file for modification, you are actually checking out a copy of a revision—usually the latest revision—from the file's revision tree. The revision you have checked out appears on your disk as an ordinary file named after its associated Projector revision tree. When you check it back in, the "file" becomes the next revision in its SourceServer revision tree.

When you intend to change a file, SourceServer gives you the opportunity to write a comment when you check it out of the database, so that other team members know that you are making alterations. SourceServer prevents other team members from making changes to the file while you have it checked out. Only a single person may change a revision at any one time. All others may

examine the revision, but they do not have the right to make changes.

You can check the revision back into the Projector database at any time, although you would normally check in revisions as soon as your modifications are complete and tested. Once your revision is checked back in, the next sequential number is, by default, appended to its name to identify its place in the revision tree. This revision then becomes available to anyone on the team.

Besides supporting a single sequence of revisions to each file, SourceServer also allows alternative sequences of revisions to be created. These alternative sequences are called branches. Branching makes possible:

- The modification of old revisions.
- Work on the same revision of a file by several programmers simultaneously.
- Parallel, experimental lines of development.

See the “Creating Branches” section later in this chapter.

Whenever you check in a file, you are encouraged to document your changes and the reasons for them. Such updates allow the database’s current status and history to be easily retrieved by all team members. The information is also extremely handy when you have to go back through old revisions to find a problem or retrieve something of value.

Projector databases may contain other projects, called subdatabases. The advantage of this is that you can break large projects into smaller ones that can still be used as a whole by those outside the immediate programming team. Subdatabases are analogous to subfolders on your disk. Subdatabases help you keep your revisions organized.

Features

Some of SourceServer’s key capabilities are listed here:

- Databases and subdatabases can be organized into a hierarchy.

- All revisions to a file are saved in the revision tree. Each revision is uniquely identified by its filename and revision number.
- Nontext files as well as text files may be stored in the database.
- Revisions made to text files are stored in a compact format.
- Access by multiple users is supported. Requests to modify the Projector database are controlled by user name on a per-project basis. AppleShare can be used to assign privileges.
- A flexible naming facility allows revisions to be identified by symbolic name as well as by filename and revision number. See the section “Symbolic names” later in this chapter.
- The entire history and status of all revision trees in the project can be displayed conveniently and accurately. A comment field is saved with revisions, revision trees, and projects. SourceServer also associates another data field, called Task, with every revision of a file.

Limitations

Keep these rules in mind when using SourceServer:

- All files (revision trees) in a Projector database must have unique names.
- There is no easy, integrated way to change a filename.
- Revisions cannot arbitrarily be deleted out of sequence. (See the section “Deleting revisions” later in this chapter.)
- Revisions to nontext files are not compressed.
- Commas are not allowed in filenames.
- Symbolic names must not be hyphenated.

See “SourceServer Database Administration” later in this chapter for information on ways to get around some of these limitations.

Note

Be careful of programs, such as Microsoft Word, that delete `'ckid'` (check ID) resources when you check it out of the database. SourceServer uses these resources to keep track of files.

Using SourceServer: A Tutorial

This section takes you through each step of the principal operations of SourceServer, demonstrating the functions of the dialogs. The concepts and organization of SourceServer will become evident as you work through this hands-on tutorial.

First you need to launch SourceServer. Be sure it resides in the `Tools` folder, which is in the folder that contains the THINK Project Manager. The default installation puts SourceServer in this folder.

The next step in the tutorial is to create a new database. After that, you check a new file into that database. Finally, you mount the project and check out a revision of one of your files.

The concepts used here are defined in detail in the section "Components of a Projector Database" later in this chapter and the summary sections following this walk-through.

Creating and mounting a new project

The simplest way to create a project is to use the **New ProjectorDB** dialog. Follow these steps to create a Projector called `BullseyeDB`:

1. Open the Bullseye project in the `Bullseye` folder. The `Bullseye` folder is in the `Demos` folder of your `Development` folder.
2. Launch SourceServer from the **Source** menu. Choose **Launch SourceServer** from the **SourceServer** hierarchical menu.
3. Select **New ProjectorDB** from the **SourceServer** menu.
4. Type `BullseyeDB` into the Database Name field of the **New ProjectorDB** dialog. Click Create.
5. You may add some descriptive information about the new project in the Comment field. Click OK.

6. Choose **Mount ProjectorDB** and mount the Projector database. in the `BullseyeDB` folder.

Note

To keep track of who checks files in and out, SourceServer requires every user to be named. It gets the name from the Owner Name field in the Sharing Setup control panel. To use SourceServer, your `Control Panels` folder must contain the Sharing Setup control panel, and you must enter a name in the Owner Name field of the Sharing Setup control panel.

The Projector database you mounted becomes the current project. `BullseyeDB` can actually be an HFS place-name, or, if a project is currently mounted, a Projector database pathname. In either case, the name of the new database is the leaf of the path. If a path is given, that directory becomes the project directory for the new project. If a Projector database pathname is given, the new Projector database becomes a subdatabase of its parent.

When you mount a Projector database, all of its subdatabases are mounted at the same time. See the section “Components of a Projector Database” later in this chapter for more information on project directories and details of naming conventions.

Checking in a revision

When you have finished modifying an existing revision of a file, you should check it back into the project. To add a new file to a project, follow the same procedure.

You can add comments to indicate the changes you’ve made. SourceServer will record the date and time, as well as other information about the revision. Your changes become part of a new revision in the file’s revision tree. (Of course, read-only revisions cannot be checked in because they do not contain changes and therefore cannot create new revisions.)

To check in the Bullseye project’s source and header files into the Projector database, choose **Check In** from the **SourceServer** menu.

Note

A header file only appears in the **Check** dialog after you compile a source file that references it.

There are buttons for selecting all your source files, header files, or files that have been modified since the last time they were checked in. To put all the files for Bullseye under source control, click All to select all the files in the Bullseye project for checking in.

Note

You can also select files by clicking their names. Use Shift-click to select contiguous filenames by dragging. Use Command-click to select discontiguous filenames.

Type a comment into the Comment field of the **Check In** dialog. Whenever you check a revision into a project, add a comment describing the changes you've made (or, in the case of a new file, its purpose). You can add comments to each file that you check in to the database. When you're done, click Check In and then Done.

Later, when you use the **Check Out** dialog, you will see that your source and header files for the Bullseye project have been entered into the Bullseye Projector database.

Opening files in a Projector database

Now open the file `Bullmenus.c` in the Bullseye project. In the lower-left corner of the `BullMenus.c` window, an icon appears of a pencil with a line through it. This means that the file has been opened as read-only and that you will not be able to make changes to it. If you click this icon, a dialog opens that displays SourceServer information about the file and that lets you edit the revision's comments.

Checking out files

In order to change a file, you need to check it out from the Projector database. If a file is currently in the frontmost window, the SourceServer menu will let you check the file out directly. By choosing **Check Out**, you can check out a group of files at once. When you check out a file or group of files, SourceServer gives you the opportunity to state what task you are performing and to enter a comment about the changes you are going to make. If other team

members want to check the file out at the same time, they will be able to see who has the file checked out and what task you are performing.

1. Choose **Check Out** from the **SourceServer** menu.

   Since the Bullseye project is already mounted, the files that you checked in earlier are visible in the dialog. The **Check Out** dialog is similar to the **Check In** dialog.

2. Select `BullMenus.c`.

   The **Check Out** dialog gives you information about the files in the database. Click the Info button. A dialog appears that displays the comment you entered in the file, the creator, and the date the file was created and last modified. If someone else had the file checked out for writing, you would also see the task they are performing on the file. Click the Back button to return to the original display.

   To make changes in `BullMenus.c`, you must check it out as modifiable. Notice that the Keep read-only radio button is enabled. If you want to edit the file, you need to open it as modifiable. To check a file out as modifiable, click the Keep modifiable radio button. Before checking out the file, remember to enter a comment and task.

3. Choose **Check Out** and then **Done**.

   You can now edit the `BullMenus.c` file. Notice the pencil icon in the lower-left corner of the `BullMenus.c` window. This means that the file is checked out for writing and that you can make modifications to the file. Since you have it checked out for writing, others will not be able to check it out for writing at the same time.

Modifying read-only files

On rare occasions, you may want to modify a read-only file. For example, you may take home a read-only copy of a file that you do not expect to modify. However, once you get home, you discover that you do, after all, need to make changes to the file. The **Modify Read-Only** menu choice from the **File** menu allows you to make these changes.

If you have a read-only file and choose this command, the file is marked as “Modify Read-Only.” This means that you can edit the file and check your changed file back into the database. If someone else has the file checked out for writing, you will be required to enter your changed file into a separate branch. This ensures that both the changes that you and the other programmer made are reflected in the database and no changes are lost.

1. Open `Bullseye.c` in the Bullseye project.

   The pencil icon in the lower-left corner of the `Bullseye.c` window has a line through it, which means the file is marked “Read-Only.”

2. Choose **Modify Read-Only** from the **File** menu.

   Now the pencil icon has a dotted line through it, which means that the file is marked “Modify Read-Only.” You can edit the file and check it into the Bullseye Projector database.

Checking in your revisions

After you make changes to files, you need to check them back in to the database.

1. Choose **Check In** from the **SourceServer** menu.

2. Click the All button to select all the files you have modified.

The Check In Dialog

The **Check In** dialog gives you many options for entering revisions into a Projector database. The sections below describe each of the dialog items and how to use them.

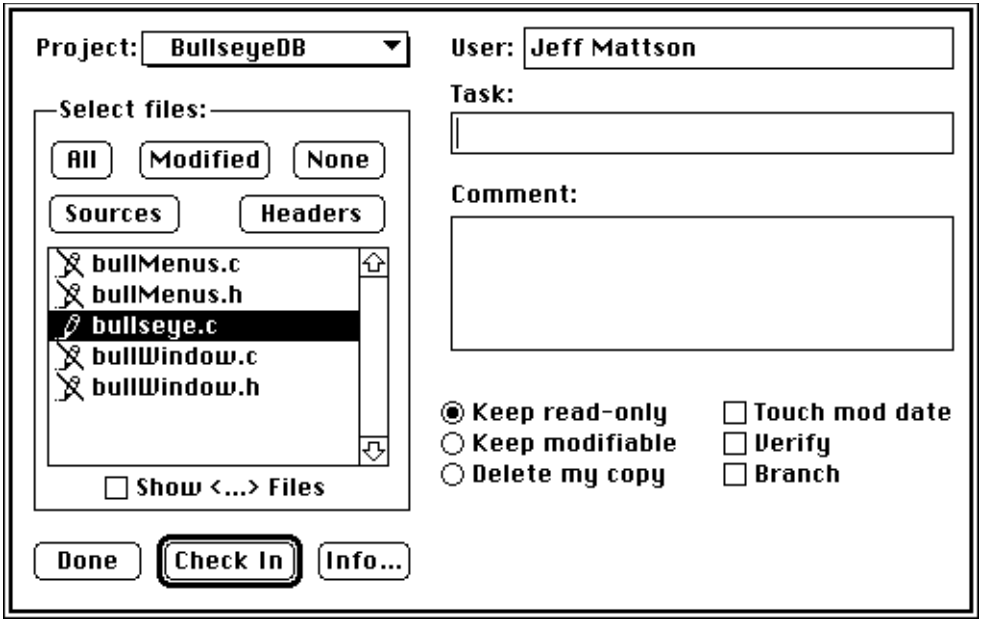

Figure 24-2 The Check In dialog

Project pop-up menu

Click the Project pop-up menu in the upper-left corner of the **Check In** dialog to select a mounted project as the current project. The current project is the project into which you will be checking files.

User field

This field usually contains your name. It must contain an entry in order for you to check in revisions. You can change the entry if you wish. The default entry is the name in the Sharing Setup control panel.

Select files area

The Select files area allows you to select the files to add to your project.

This button…	Does this…
All	Selects all of the files in the current project
Modified	Selects files that have been modified since being checked out of the database
Sources	Selects source files (`.c`, `.cp`, `.cpp`, and `.r`) in the current project
Headers	Selects header files (`.h`, `.hp`, and `.hpp`) in the current project
None	Deselects any selected files

Table 24-1 Buttons in the Select files area

By default, source and header files that exist in the THINK tree are not shown. To show those files as well, select the Show <...> Files option.

Task and comment fields

You can enter a brief description in the Task field. You can also add comments to indicate the changes you've made. SourceServer records the date and time, as well as other information about the revision. Your changes become part of a new revision in the file's revision tree.

Keep read-only, Keep modifiable, and Delete my copy buttons

The default is a read-only copy of a file. By using the three radio buttons in the **Check In** dialog, you can alter this default. The top button corresponds to the default; clicking it leaves you with a read-only copy. Use the Keep modifiable button to check in the file and still retain a modifiable copy. The Delete my copy button deletes your copy of the file once it's successfully checked in.

Touch mod date check box

This option appears in both the **Check In** and the **Check Out** dialogs. In both cases, it lets you change the convention for time and date stamping. In the **Check In** dialog, Projector's default is to leave the date and time of check-in untouched. Check the Touch mod date

box to stamp the revision of the file that you are checking in with the current date and time.

Verify check box

The Verify check box asks Projector to verify the contents of files to ensure that file corruption has not occurred, such as during file transfers.

Check In button

The Check In button tells SourceServer to add the selected files to the Projector database.

Done button

When you are finished checking in files, click Done to leave the **Check In** dialog.

The Check Out Dialog

Choosing **Check Out** from the **SourceServer** menu brings up the **Check Out** dialog, which gives you the opportunity to check out files from a Projector database.

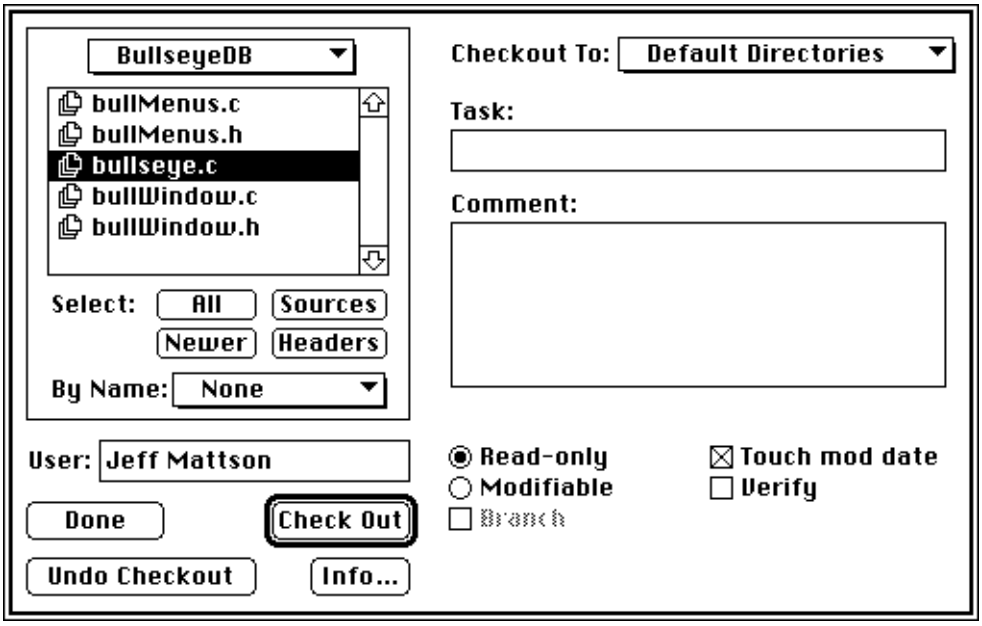

Figure 24-3 The Check Out dialog

Select files area

The Select files area allows you to select files to check out of the Projector databases. The pop-up menu at the top of this area allows you to choose from the mounted projects. Click the pop-up menu to select a mounted project.

Below the pop-up menu is a list box that displays the available files. To refer to particular revisions of a file, double-click the filename to display its revision tree. You can then select and act upon individual revisions.

To the left of each filename and revision is an icon that represents the current status of the file in the database. Table 24-2 lists the possible icons.

Icon	Description
	Project. A similar, but larger icon is used in the Finder to represent the `ProjectorDB` file.
	Revision tree. This icon appears only in the **Check Out** dialog.
	Single revision. The regular document icon represents an individual revision that's currently available. It is visible when an individual revision tree is displayed.
	Pencil. When a project is displayed (so that all its revision trees are listed), the pencil icon means that the latest revision of the main trunk is checked out for modification by the current user. When an individual revision within a revision tree is displayed (a list of revisions), the pencil icon means that the particular revision is checked out for modification by the current user.
	Lock. When a project is displayed (so that all its revision trees are listed), the lock icon means that the latest revision of the main trunk is checked out for modification by another user. When an individual revision within a revision tree is displayed (a list of revisions), the lock icon means that the particular revision is checked out for revision by another user.

Table 24-2 File status icons

Select buttons
There are four Select buttons that allow you to choose sets of files:

All
If you are checking revisions out for modification, the All button selects all revision trees whose most recent revisions are not already checked out for modification. If you are checking out read-only copies, the All button selects all revisions in the project.

Newer
Use the Newer button to select all the revisions for which the latest revision on the main trunk is not in the Checkout To directory. Projector looks in the checkout directory to determine which revisions to compare against the project. If you have a modifiable file (or a file that is on a branch) in the checkout directory, then the corresponding revision trees are not selected. In the case of modifiable files, this is done to prevent your checking out a file by overwriting a modifiable file. In the case of any files that you may have on a branch, Projector assumes that you want to leave those files alone.

Sources
Selects source files (with names ending in `.c`, `.cp`, `.cpp`, and `.r`) in the current project.

Headers
Selects header files (with names ending in `.h`, `.hp`, and `.hpp`) in the current project.

Note

If you have read-only copies of branches, then use the Select All button to ensure that you get the latest revision on the main trunk of every revision tree.

If you hold down the Option key as you click the Newer button, then revisions that are new to the project (that is, revisions of which you don't already have a copy) will not be selected.

The Newer button cannot be used when you are checking out revisions for modification.

Note

The two buttons Select All and Select Newer do not actually check out revisions; they simply make a selection in the project list (the leftmost frame of the **Check Out** dialog). Only the Check Out button checks out the selected revisions.

By Name field

Click the By Name field to display a pop-up menu showing both private and public symbolic names. The private names appear at the top, the public at the bottom. When you select a symbolic name, the project list at the left side of the dialog highlights all revision trees that correspond to the selected symbolic name. Click the Check Out button to checkout all of them. This makes it easy to select just those revisions that compose, for example, an alpha release.

Note

If you manually change the selection after selecting a symbolic name, you will cancel the selection of the name.

User field

This field usually contains your name. It must contain an entry for you to check out revisions. You can change the entry from the default (the Chooser name).

Done button

When you're finished checking out files, click Done to leave the **Check Out** dialog.

Check Out button

The Check Out button tells SourceServer to check the selected files out from the Projector database.

Undo Checkout button

To undo any changes, use the Undo Checkout button located in the lower-left corner of the dialog. For example, if half of the revision trees were accidentally checked out for modification, you could undo the mistake by simply canceling their checkouts.

Info button

If you have selected an individual revision in the revision list, clicking the Info button overlays the right side of the **Check Out** dialog with the Info View, which is a display of information pertinent to the selected revision. At this point you can edit the Task and Comment fields. You can also get information about revisions that are checked out for modification or about any other revision in the project.

Checkout To directory field

The Checkout To field in the dialog's upper-right corner shows the directory where checked-out revisions will be placed. Clicking the field displays a pop-up menu that gives you three choices:

- The checkout directory for the current project
- The current directory
- Access to a standard file dialog, where you can choose any directory

The default directory that is displayed in the field is the checkout directory for the current project.

Task and Comment fields

The Task and Comment fields are optional, but you should get in the habit of always stating your purpose. Use the Task field to let others know why you have checked out the revision. This information can help later if you have to review a long history to find a particular change or edit.

The Comment field is intended to document the specific changes to a revision, while the Task field could be used to relate different revisions, perhaps across several files (that is, Projector revision trees). You can change the purpose stated in the Task and Comment fields when you check the revision back in.

For example, implementing a certain feature might require several changes to each of three files. Each revision might have a different comment, but the tasks for all the revisions might say “Enhancement X.” The Task field makes it easier to look at the history of a project and determine which changes were made to accomplish various tasks.

To save a Task field across several revisions, select the revisions in the list at the left of the dialog and type directly in the Task field of

the **Check In** dialog. Then click the Check In button to check in the selected files. To save a unique Comment field with a particular revision, first select that revision and then click the Info button. Type in the Info View's Comment field, click Save, then click Done.

Read-only and Modifiable buttons

The two radio buttons at the bottom of the dialog specify read-only or write-modify types of revision checkout. The default is to check out read-only copies. Everyone with access to the project can check out revisions for read-only at any time. Only one person at a time, however, may check a file out for modification.

Branch check box

Use this check box to create a branch. This control is enabled only when you are doing modifiable checkouts. If the file that you have selected is locked, then this check box will be selected automatically. See the section "Creating Branches" later in this chapter for instructions on creating branches, and see the section "Components of a Projector Database" later in this chapter for a detailed explanation of branching in Projector.

Note

You can select dimmed files by holding down the Option key while you click the name of the files.

Touch mod date

When the Touch Mod Date option is selected, the revisions checked out (into the Checkout To directory) are touched—that is, the modification date is set to the date and time of checkout. If the box isn't checked, the checked-out revisions have the same modification date as when the corresponding revision was checked in.

Verify check box

The Verify check box asks Projector to verify the contents of files to ensure that file corruption has not occurred, such as during file transfers.

The Name Revisions Dialog

Frequently you need to work on the same group of files for a long period of time. The **Name Revisions** command lets you create a named set of files that you can easily check out all at once. When you choose the name of the revision set from the By Name pop-up menu in the **Check Out** dialog, SourceServer selects all that set's files in the Select files area.

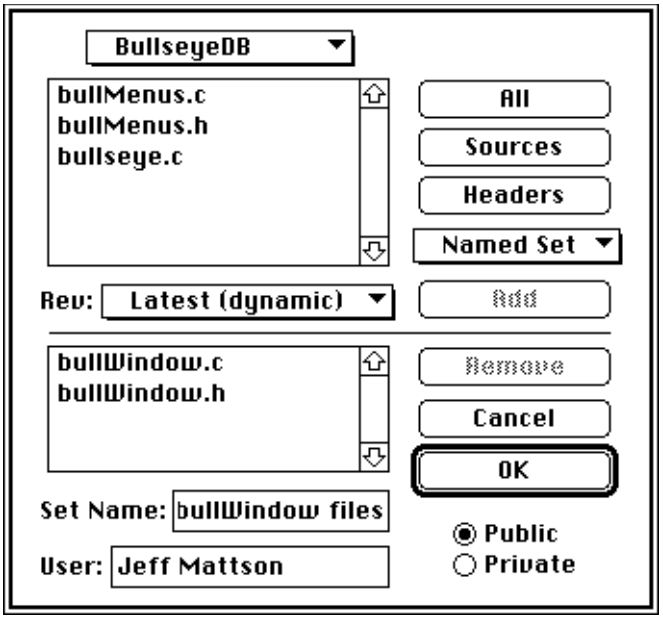

Figure 24-4 The Name Revisions dialog

You can control which file versions the revision set selects with the Rev pop-up menu. To have the set select the most current file in the revision tree, choose Latest (dynamic). For versions of the files that are current when you name the revision, choose Current (static). This option is especially useful when you want a named set for a frozen version of code, such as beta or final releases.

Creating Branches

In addition to supporting a sequence of revisions to a file in a project, SourceServer lets you create branches. Branches are alternative sequences of revisions that are parallel to the main revision sequence.

Branching makes possible:

- Modification of old revisions
- Work on the same revision of a file by several programmers simultaneously
- Parallel, experimental lines of development

You can create a branch revision during the checkout or checkin process by clicking the Branch check box in the **Check In** or **Check Out** dialogs.

Checking out a modifiable copy of an old revision creates a new branch. When the revision is checked back in, it automatically becomes a branch from the original revision.

Components of a Projector Database

This section explains in detail how SourceServer works. SourceServer keeps track of these components for each project under its supervision:

- Projector databases
- Nested databases (subdatabases)
- The files (revision trees) belonging to a SourceServer database
- All revisions of each file (revision tree)
- The branches of each file (revision tree)
- Names of every user creating or modifying a file or database
- Symbolic names

Note

Most of the actions described in this chapter can be performed both directly from dialogs within the THINK Project Manager and by typing the command line into a THINK Editor window, selecting the command, and pressing Control-Enter. Some of these operations can only be performed by the command line method. For more information, see the section “Using the Command-Line Interface” later in this chapter.

Projector database

A Projector database consists of a project name, an author, some text describing the project, a set of revision trees belonging to the database, and whatever subdatabases the project may have, which are actually projects in their own right. The author is the person who created the project.

Projector databases can reside locally on an individual user’s disk, or they can be placed on a file server to facilitate access by multiple users. AppleShare can be used to assign access privileges to various users. SourceServer does not itself provide facilities for assigning access privileges.

Use the Project pop-up menu in either the **Check In** dialog or the **Check Out** dialog to set the current project from the currently mounted Projector databases. SourceServer assumes that all SourceServer commands pertain to the current project unless told otherwise. To mount other databases, choose **Mount ProjectorDB** from the **SourceServer** menu.

The project directory is the directory in which a given project resides. It is defined at the time the project is created. All revisions to all revision trees and all other SourceServer information are kept in the project directory within the project file, called ProjectorDB. All users access the same ProjectorDB. Nested databases are also kept in this directory as subdirectories.

Every user has a checkout directory for each project. This is the directory into which, by default, Projector places checked-out revisions. You can change the checkout directory by using the Checkout To pop-up menu in the **Check Out** dialog.

Each user can select one or more projects for access by choosing **Mount ProjectorDB** from the **SourceServer** menu. Selecting a project makes it and all its nested projects (subprojects) accessible to the user. You can remove projects from the project list by choosing **Unmount Project** from the **SourceServer** menu.

The location of the project directory is the same for everyone, but the checkout directory can be different for each individual.

Revision trees

When a file is added to a project, it forms the first revision of a new revision tree. The new revision tree bears the original file's name. Each revision tree in a project consists of the following component:

- The file name
- The name of the author—the person who added the original file to the project
- A comment describing the revision tree
- A record describing the current state of the revision tree—that is, who has checked out the revision tree and all of its revisions
- The set of revisions and branches of the original file

SourceServer can be used with all types of files, such as source files, libraries, applications, and so on. The only difference between text files and nontext files is that revisions to nontext files are not compressed, and they cannot be automatically opened at checkout time. You can check out read-only copies of nontext files, or check out any such revision for modification and then check it back in as a new revision. Revisions of nontext files can also be named and deleted.

Each time a programmer checks in an updated copy of a revision, a new revision is created. As changes are made and the number of revisions grows, a revision tree forms. The revision tree traces the history of the file. By accessing various portions of this tree, you can retrieve, inspect, and compare any of the previous revisions of a file. SourceServer also allows old revisions to be deleted when they are no longer of interest, provided that you delete all old revisions prior to a selected revision. In other words, arbitrary deletion is not allowed.

Once a revision is checked out for modification, it is locked, thus preventing a second modifiable copy of that revision from being checked out at the same time. You may check out a read-only copy of the locked revision.

However, it is possible to check out a modifiable copy of another revision. If you insist on checking out a revision that is already locked, SourceServer creates a branch for this new copy. You can later merge the changes to synchronize the file. Each revision in a revision tree contains the following identification:

- The revision number
- The creation date
- A task field describing any specific tasks undertaken or any other information
- A comment field describing the reason for the revision
- The name of the author of the revision
- A copy of the revision itself (compacted, in the case of source or text files)

Revisions are normally identified in numeric order (1, 2, 3,…99, 100, 101, and so on). However, if you are using the command-line interface described later in this chapter, you can use major/minor numbering instead (1.1, 1.2, 1.3,…1.99, 1.00, 1.101,...2.1, 2.2, and so on). When a new revision is checked in, SourceServer automatically increases its revision number by 1 (for example, from 4 to 5, or 4.9.2 to 4.9.3). You can override this action by specifying a different revision number. The only restriction is that this new number must be sequentially greater than the revision that was checked out. Here's the syntax for major/minor number:

Major[`.`*Minor*]

To specify a particular revision in a command, append a comma followed by the desired revision number to the end of the name. In other words, `file.c,3` refers to the third revision of `file.c`. (Remember that commas are not allowed in project filenames.) The first command in the following example checks out the latest (current) revision of `file.c`. The second command checks out revision 3 of `file.c`, regardless of what the current revision is.

```
CheckOut file.c
CheckOut file.c,3
```

The following command checks in `file.c`, forcing the revision to 4.1:

```
CheckIn file.c,4.1
```

This command is legal only if the revision that was checked out was less than 4.1, such as 4, 3.9, 4.0.9, 2, and so on.

Note

You can only use the major/minor numbering convention with the command-line interface to SourceServer.

Branches

In addition to supporting a sequence of revisions to a file in a project, SourceServer also lets you create branches. Branches are alternative sequences of revisions that are parallel to the main revision sequence.

To refer to particular revisions when using the **Check Out** command, double-click a filename to display its revision tree. You can then select and act upon the individual revisions to check out a particular revision of a file or get information about that revision.

Symbolic names

SourceServer supports a general-purpose naming facility, the **Name Revisions** command, that allows project users to easily specify revision trees, versions, and branches within a project. For more information, see the section “The Name Revisions Dialog” earlier in this chapter.

SourceServer Database Administration

The administrative duties for projects under SourceServer are very simple. Anyone who has write access to the database (under AppleShare) can administer the project.

Responsibilities include:

- Creating nested databases
- Moving, renaming, and deleting projects
- Deleting old revisions that are no longer needed
- Renaming a file in a project
- Deleting files that do not belong in the project
- Removing Projector information from files

Creating nested databases

If your Projector database is especially large—over 50 files—performing such simple operations as bringing up the **Check Out** dialog could take an unacceptably long amount of time. And if the database is on a file server, you could start having bad performance with only 25 files. The best way to avoid waiting is to break up your database into nested databases.

To create nested databases, first create your top-level database. Then create each of your nested databases inside the top-level database's folder. Figure 24-5 shows the top-level database `MiniEditDB`, which holds the five nested databases `mainDB`, `fileDB`, `printDB`, `windowDB`, and `utilDB`.

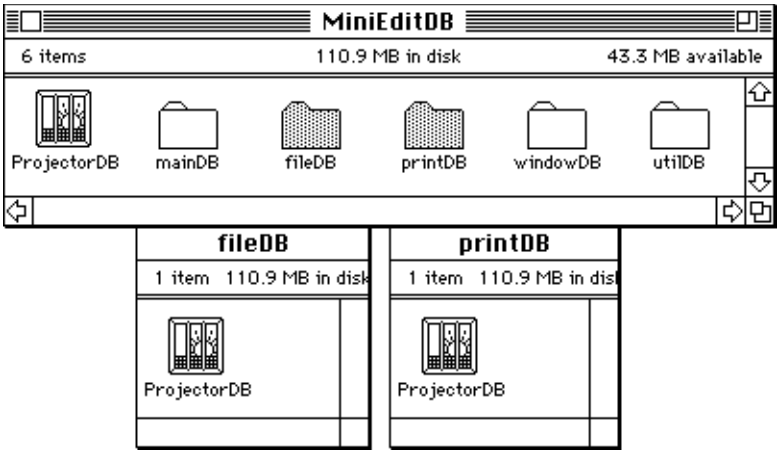

Figure 24-5 Nested databases

Whenever you mount a top-level database, SourceServer automatically mounts its nested databases. In any SourceServer dialog that lists the mounted databases in a pop-up menu, you'll see each nested database labeled with the top-level database's name and

the nested database's name, separated by the character ∫, as shown in Figure 24-6.

Figure 24-6 Viewing nested databases in a pop-up menu

Moving, renaming, and deleting projects

You can move or rename a project by using the Finder. Simply renaming the database directory renames the project.

There are two points to keep in mind when moving or renaming a project:

- The project hierarchy changes. Command line Mount-Project commands must be modified to reflect the new location of the project.
- Projects should be moved or renamed only when no revisions are checked out for modification. Furthermore, after the project has been changed, *all* read-only copies should be checked out again. SourceServer puts the project name in the resource forks of revisions during checkout. Once the project is moved or renamed, that information is no longer valid.

You can delete an entire project by deleting the folder containing the project.

Warning

Once you delete the project, all files and their revisions are lost, unless you've checked them out to another folder first.

Deleting revisions

You can delete old revisions in a revision tree by using the command-line operation `DeleteRevisions` and specifying the oldest revision that you want to keep. All prior revisions are deleted.

For example, if you specify the revision `filename.c,5`, then `filename.c`'s revisions 4, 3, 2, and 1 are deleted, leaving `filename.c,5` intact. You cannot, however, arbitrarily delete `filename.c,4` while leaving revisions 3, 2, and 1 intact. This restriction prevents confusion of the revision numbering scheme.

You can delete entire branches by naming the branch (for instance, `filename.c,22a`).

Warning

Once you delete revisions, you cannot recover them.

If you accidentally check a file into the wrong project, you can remove it and all its revisions by using the `-file` option of the `DeleteRevisions` command.

Removing a file from a project

After you check in a file to a SourceServer database, the file contains a `'ckid'` resource that stores project information and that prevents you from checking in the file to a different database. To remove any association a file has with SourceServer, use the `OrphanFiles` command described at the end of this chapter.

Note

Once you use the `OrphanFiles` command, you cannot check the file back into its former project. You can, however, use the `TransferCkid` command, described at the end of this chapter, to transfer a `'ckid'` resource with the correct project information to the file.

Using the Command-Line Interface

In addition to using the dialogs available through the **Source** menu, you can send commands to SourceServer directly through the SourceServer command-line interface. The commands and options are the same as MPW SourceServer commands, with the following exceptions:

- The commands `MergeBranch` and `CompareRevisions` are not available.

- Commands that have `-w` and `-close` options in MPW do not have corresponding options in SourceServer.

You can enter the commands in any THINK Editor window. Just enter the command and choose **Send Command** from the **SourceServer** hierarchical menu or press Control-Enter. The THINK Project Manager sends the command to the SourceServer and prints the results in the edit window.

Here are some notes to keep in mind as you use the SourceServer command-line interface:

- To specify a filename, use its full pathname.
- When you use the `DeleteRevisions` command, be sure to specify the `-y` option.
- The `help` command is not available, unless you have ToolServer.
- SourceServer commands, filenames, and folder names are *not* case sensitive. For example, SourceServer treats the names `CheckIn` and `checkin` alike.
- When you enter a file or folder name with a space in it, you must enclose the entire name in double quotes. For example, your must refer to the file `My File` as `"My File"`, but you can refer to the file `File1` as just `File1`.
- You can specify a file on the Desktop like this: "*HD Name*`:Desktop Folder:`*filename*".

SourceServer Command Descriptions

In all the following command descriptions and syntax listings, optional parameters are enclosed in square brackets ([]). Parameters separated by the vertical bar (|) are mutually exclusive parameters.

CheckIn

```
CheckIn  [-p] [-u user] [-project project] [-t task]
         [-new | -b] [-cs comment | -cf file]
         [-m | -delete] [-touch] [-verify]
         [-y|-n|-c] (-a | file...)
```

Checks a file or files into a projector database.

Option	Description
`-a`	Check in all files in the current directory
`-b`	Check in *file* as branches
`-c`	Cancel if conflict occurs (avoids dialog)
`-cf` *file*	Look for comments in the file
`-cs` *comment*	Add a comment, which should be a description of changes made to the file
`-delete`	Delete the file after checking it in
`-m`	Check out the files for modification after checking in
`-n`	Answer no to all dialogs (avoids dialogs)
`-new`	Add a new file to the project
`-p`	Write progress information to the edit window below the command you entered
`-project` *project*	Check in the files to *project*
`-t` *task*	Add a short description of task accomplished
`-touch`	Update the modification date of the files after checking in
`-u` *user*	Check in the files under the user name *user*
`-verify`	Verify the contents of the checked-in file
`-y`	Answer yes to all dialogs (avoids dialogs)

Table 24-3 Options for the CheckIn command

CheckOut

```
CheckOut[-p] [-m] [-b] [-cancel] [-u user]
        [-project project] [-t task] [-d dir]
        [-cs comment | -cf file] [-r]
        [-y|-n|-c] [-noTouch | -touchOnly]
        [-history] [-verify] [-obsolete]
        (-update | -newer | -a | name...)
```

Checks out a file or files from a projector database.

Option	Description
`-a`	Check out all the files in the current project
`-b`	Check out specified files on a new branch
`-c`	Cancel if conflict occurs (avoids dialog)
`-cancel`	Cancel the checkout of the files
`-cf` *file*	Look for the comment in the file
`-cs` *comment*	Add your comment, which should be a short description of changes
`-d` *dir*	Name the directory where the checked-out files should go
`-history`	Store all the project history information, including the author, checkin date, task, and comment, in the `'ckid'` resource for the checked-out files
`-m`	Check out a modifiable copy of the file
`-n`	Answer no to all dialogs (avoids dialogs)
`-newer`	Check out the latest copy of all files in the project
`-noTouch`	Don't touch the modification date of the checked-out files
`-obsolete`	Check out a read-only version of an obsoleted file

Table 24-4 Options for the CheckOut command

Option	Description
`-p`	Write progress information to the edit window below the command you entered
`-project` *project*	Name of project that contains the files
`-r`	Recursively check out the files in the project and all its subprojects
`-t` *task*	Add a short description of the task accomplished
`-touchOnly`	Check out and reset the modification data of only those members of a named set that are not already in the checkout directory
`-u` *user*	Give the name of current user
`-update`	Check out the latest copy of all files you already have
`-verify`	Verify the contents of the checked-out file
`-y`	Answer yes to all dialogs (avoids dialogs)

Table 24-4 Options for the CheckOut command *(Continued)*

Note

Name can be the name of a file or the name of a set created with the **Name Revisions** command.

CheckOutDir

```
CheckOutDir [-project Project | -m] [-r]
            [-x | directory]
```

Specifies the directory where checked-out files will be placed.

Options	Description
`-m`	List the checkout directories of all root projects
`-project` *project*	Name the project to associate with the checkout directory

Table 24-5 Options for the CheckOutDir command

Options	Description
`-r`	Recursively set or display the checkout directories
`-x`	Reset the checkout directories to `":"`

Table 24-5 Options for the CheckOutDir command *(Continued)*

CompareRevisions Not available with SourceServer.

DeleteNames

```
DeleteNames [-u user] [-project project]
            [-public | -private] [-r]
            [name... | -a]
```

Deletes user-defined symbolic names.

Option	Description
`-a`	Delete all names
`-private`	Delete private name
`-project` *project*	Name of project that contains the file
`-public`	Delete public names (the default)
`-r`	Delete names recursively
`-u` *user*	Name of current user

Table 24-6 Options for DeleteNames

DeleteRevisions

```
DeleteRevisions   [-u user] [-project project]
                  [-file] -y file...
```

Deletes previous revisions of files in a project.

Option	Description
`-file`	Delete the file and all its revisions
`-project` *project*	Name of the project that contains the files
`-u` *user*	Delete the names of the current users
`-y`	Delete the file/revision

Table 24-7 Options for DeleteRevisions

Warning

Use the `-y` option. If you don't, SourceServer will put up a dialog box that you cannot respond to.

MergeBranch Not available with SourceServer.

ModifyReadOnly

`ModifyReadOnly` *file*

Lets you edit a read-only Projector file.

MountProject

`MountProject` `([-s] [-pp] [-q] [-r]) | [`*project* `[-only] ]`

Mounts projects.

Option	Description
`-only`	Mount only *project* and none of its subprojects
`-pp`	List mounted projects using project paths
`-q`	Don't quote names with special characters
`-r`	Mount the project and all its subprojects recursively
`-s`	Print names only, not commands

Table 24-8 Options for MountProject

NameRevisions

```
NameRevisions  [-u user] [-project project]
               [-public | -private | -b] [-r]
               [[-only] | name
                [[-expand] [-s] |
                 [-replace] [-dynamic]
                 [names... | -a]]]
```

Defines a symbolic name.

Option	Description
`-u` *user*	Name the current user
`-private`	Create a private name
`-project` *project*	Name the project that contains the revisions
`-public`	Create a public name (the default)
`-b`	Print both public and private names

Table 24-9 Options for NameRevisions

Option	Description
`-expand`	Evaluate names to revision level before printing
`-only`	Only print the names, not the associated revisions
`-replace`	Completely overwrite the previous definition of name
`-dynamic`	Evaluate names to revision level when using not defining
`-r`	Recursively execute `NameRevisions` on the current project and all its subprojects
`-s`	Print a single name per line
`-a`	Create named revision containing all files in the project

Table 24-9 Options for NameRevisions *(Continued)*

NewProject

`NewProject [-u` *user*`] [-cs` *comment* `| -cf` *file*`]` *project*

Creates a new project.

Option	Description
`-cf` *file*	Look for the comments in file
`-cs` *comment*	Add a short description of the project
`-u` *user*	Name the current user

OrphanFiles

`OrphanFiles` *file*...

Removes Projector information (`'ckid'` resource) from a list of files.

Project

`Project [-q |` *project* `]`

Sets or writes the current project.

Option	Description
`-q`	Don't quote the names of projects with special characters

ProjectInfo

```
ProjectInfo [-project project] [-log] [-comments]
            [-latest] [-f] [-r] [-s]
            [-only | -m] [-af author | -a author]
            [-df dates | -d dates]
            [-cf pattern | -c pattern] [-t pattern]
            [-n name] [-update | -newer] [file...]
```

Displays information about a project.

Option	Description
`-a` *author*	Only list revisions created by *author*
`-af` *author*	Only list files created by *author*
`-c` *pattern*	Only list revisions whose comment contains *pattern*
`-cf` *pattern*	Only list files whose comment contains *pattern*
`-comments`	List comments along with the rest of the information
`-d` *dates*	Only list revisions whose creation date is within *dates*
`-df` *dates*	Only list files whose modification date is within *dates*
`-f`	List file information
`-log`	Print project log
`-m`	Only list files/revisions that are checked out
`-newer`	Give information on files that would be checked out using this option
`-only`	Only list project information
`-project` *project*	Name the project to get information on
`-r`	Recursively list the project and all its subprojects

Table 24-10 Options for ProjectInfo

Option	Description
`-latest`	Only list information on the latest revision on the main trunk
`-s`	Use short listing, names, and revision names only
`-t` *pattern*	Only list revisions whose task contains the *pattern*
`-n` *name*	Only list revisions that have *name*
`-update`	Give information on files that would be checked out using this option

Table 24-10 Options for ProjectInfo *(Continued)*

Pattern is either a literal string, like `foo`, or a regular expression enclosed in slashes, like `/foo*/`.

Date can be one of the following, where date is *mm/dd/yy* `[[`*hh*`:`*mm*`[:`*ss*`]] [AM|PM]]`:

Option	Description
date	On date
<*date*	Before but not including date
≤*date*	Before and including date
>*date*	After and not including date
≥*date*	After and including date
date`-`*date*	Between and including dates

Name may take the following forms:

Option	Description
name	In name
<*name*	Before name
≤*name*	Before and including name
>*name*	After name
≥*name*	After and including name

TransferCkid

`TransferCkid` *fromFile toFile*

Moves Projector information from one file to another.

UnmountProject

`UnmountProject [-a |` *Project…*`]`

Unmounts project.

Option	Description
`-a`	Unmount all mounted projects